Integrating CSF Components with Automation and Change Management for Effective Cybersecurity Implementation

1. How would you apply the cybersecurity framework (CSF) components and functions?
2. How would you utilize automation and change management to facilitate this process effectively?

Answer:

Integrating CSF Components with Automation and Change Management for Effective Cybersecurity Implementation In China

The NIST Cybersecurity Framework (CSF) must be adapted for China by bringing its fundamental functions—Identify, Protect, Detect, Respond, and Recover—in line with regional laws (such as the Cybersecurity Law and Data Security Law) and cultural standards. This entails deploying monitoring technologies authorised by the surveillance legislation, establishing access restrictions that adhere to hierarchical structures, cataloguing assets in accordance with data localisation laws, and coordinating incident response with the Cybersecurity Administration of China (CAC). Through AI/ML-driven threat detection, compliance monitoring in line with Chinese standards (such as GB/T 22239-2019), and SOAR tools for quick incident response, automation improves this process. To guarantee organisational buy-in, change management techniques place a high priority on feedback via platforms like WeChat, Mandarin-tailored training programs, and stakeholder interaction with local authorities. The CSF is further integrated into the national cybersecurity ecosystem by partnerships with domestic tech giants (like Alibaba Cloud) and alignment with China's Multi-Level Protection Scheme (MLPS 2.0), which strikes a balance between localised implementation and international best practices.

Code:

```
# -*- coding: utf-8 -*-
"""
NIST CSF Adaptation for China Compliance
Aligns with Cybersecurity Law, Data Security Law, MLPS 2.0, and
cultural norms.
"""

import logging
from typing import Dict, List
from datetime import datetime

# Local compliance libraries (hypothetical)
from china_compliance import DataLocalization, MLPS20Validator
from cac_integration import CACIncidentReporter
from alibaba_cloud import AlibabaSecurityHub
from wechat_api import WeChatFeedback

class ChinaCybersecurityFramework:
    def __init__(self):
        self.assets = {}
        self.access_controls = {}
        self.monitoring_tools = []
        self.incident_response_plan = {}
        self.recovery_strategy = {}
        self.compliance_log = []

        # Initialize local compliance modules
        self.data_localization = DataLocalization()
        self.mlps_validator = MLPS20Validator()
        self.cac_reporter = CACIncidentReporter()
```

```
        self.alibaba_hub = AlibabaSecurityHub()
        self.wechat_feedback = WeChatFeedback()

    # NIST CSF Core Functions
    def identify(self) -> None:
        """Catalog assets under data localization rules (Cybersecurity
Law Article 37)"""
        logging.info("Starting asset identification with data
localization checks")
        self.assets = self.data_localization.scan_and_classify_assets()

        # MLPS 2.0 alignment
        self.mlps_validator.assign_protection_levels(self.assets)

    def protect(self) -> None:
        """Implement hierarchical access controls (GB/T 22239-2019)"""
        logging.info("Applying role-based access controls with
organizational hierarchy")
        self.access_controls = self._generate_hierarchical_policies()

        # Encryption requirements
        self.data_localization.enforce_encryption(self.assets)

    def detect(self) -> None:
        """Deploy government-approved monitoring tools"""
        logging.info("Initializing surveillance-law-compliant
monitoring")
        self.monitoring_tools = [
            "Approved_IDS_System",
            "ML-Based_Anomaly_Detector"
        ]
```

```
        # AI/ML threat detection (hypothetical)
        self._start_ai_monitoring()

    def respond(self, incident: Dict) -> None:
        """Coordinate response with CAC (Cybersecurity Law Article
51)"""
        logging.warning(f"Incident detected: {incident['type']}")
        self.cac_reporter.notify_authorities(incident)

        # SOAR automation
        self._execute_response_playbook(incident)

    def recover(self) -> None:
        """Restore operations using localized backup systems"""
        logging.info("Initiating recovery protocols")
        self.data_localization.restore_from_backups()
        self._update_recovery_strategy()

    # Automation & Compliance
    def _start_ai_monitoring(self) -> None:
        """AI/ML-driven threat detection aligned with Chinese
standards"""
        # Hypothetical ML model integration
        from compliance_ml import ThreatDetector
        detector = ThreatDetector(standard="GB/T 22239-2019")
        detector.train_on_local_threat_data()

    def check_compliance(self) -> List[str]:
        """Automated compliance checks against Chinese standards"""
        return self.mlps_validator.validate(self.assets,
self.access_controls)
```

```
    # Change Management
    def conduct_training(self) -> None:
        """Mandarin-tailored training programs"""
        training_modules = {
            "data_security": "data_security_training_mandarin.mp4",
            "incident_response": "incident_response_protocol.pdf"
        }
        self.wechat_feedback.distribute_training_materials(training_mod
ules)

    def collect_feedback(self) -> None:
        """Stakeholder feedback via WeChat"""
        feedback = self.wechat_feedback.get_employee_feedback()
        self._update_policies_based_on_feedback(feedback)

    # Partnership Integration
    def integrate_alibaba_services(self) -> None:
        """Leverage Alibaba Cloud security solutions"""
        self.alibaba_hub.enable_advanced_protection()
        self.alibaba_hub.sync_with_mlps_20()

    # Helper methods
    def _generate_hierarchical_policies(self) -> Dict:
        # Implement role-based access control respecting organizational
hierarchy
        return {
            "admin": ["full_access"],
            "manager": ["department_data"],
            "employee": ["restricted_access"]
        }

    def _execute_response_playbook(self, incident: Dict) -> None:
```

```
        # SOAR integration for automated response
        pass

    def _update_recovery_strategy(self) -> None:
        # Update strategy based on post-incident analysis
        pass

    def _update_policies_based_on_feedback(self, feedback: List) ->
None:
        # Policy adjustment mechanism
        pass

if __name__ == "__main__":
    # Initialize framework
    csf = ChinaCybersecurityFramework()

    # Core CSF functions execution
    csf.identify()
    csf.protect()
    csf.detect()
    csf.integrate_alibaba_services()

    # Simulate incident response
    test_incident = {
        "type": "data_breach",
        "severity": "high",
        "timestamp": datetime.now()
    }
    csf.respond(test_incident)
    csf.recover()

    # Compliance and change management
```

```
    compliance_issues = csf.check_compliance()
    csf.conduct_training()
    csf.collect_feedback()
```

Integrating CSF Components with Automation and Change Management for Effective Cybersecurity Implementation In United State Of America

To handle changing threats, the U.S. cybersecurity plan for 2020–2025 combines automation, change management, and the NIST Cybersecurity Framework (CSF). The five CSF functions—Identify, Protect, Detect, Respond, and Recover—are designed to address U.S. priorities, including secure cloud-based recovery following ransomware attacks, zero-trust adoption for remote work, AI-driven threat detection for IoT/OT systems, and critical infrastructure protection (energy, healthcare, and finance). This approach is improved by automation through automated patch management to reduce vulnerabilities in older systems, AI/ML for predictive analytics, and SOAR platforms for incident response. Through worker upskilling, stakeholder participation, and open communication throughout changes, change management guarantees smooth adoption. The United States gets more resilience against ransomware spikes, shorter incident response times, and a cultural change towards AI-driven security practices by integrating automation with CSF operations (e.g., automated rollback for recovery, machine learning for threat prioritisation). In order to protect digital ecosystems, this strategy strikes a compromise between strategic governance and tactical innovation, as supported by federal compliance requirements (CMMC, NIST SP 800-53) and lessons learnt from events like SolarWinds.

Code:

```
import logging
```

```
from abc import ABC, abstractmethod
from typing import Dict, List

# Configure logging for incident tracking
logging.basicConfig(level=logging.INFO, format='%(asctime)s -
%(levelname)s - %(message)s')

class NIST_CSF(ABC):
    """Abstract base class for NIST CSF functions with automation and
change management."""

    @abstractmethod
    def execute(self):
        pass

    @abstractmethod
    def automate(self):
        pass

    @abstractmethod
    def manage_change(self):
        pass

class Identify(NIST_CSF):
    """IDENTIFY: Asset management and risk assessment with AI-driven
asset discovery."""

    def __init__(self):
        self.assets = []
        self.risk_scores = {}

    def execute(self):
```

```
        self.automate()
        self.manage_change()

    def automate(self):
        """Automates asset discovery and risk scoring using AI/ML."""
        logging.info("Running AI-driven asset discovery...")
        # Simulate asset discovery (e.g., IoT/OT devices in critical
infrastructure)
        self.assets = ["Energy_Controller_01", "Hospital_EHR_System",
"Banking_API_Gateway"]
        # Simulate ML-based risk scoring
        self.risk_scores = {asset: round(abs(hash(asset)) % 100) for
asset in self.assets}
        logging.info(f"Risk Scores: {self.risk_scores}")

    def manage_change(self):
        """Engages stakeholders and updates risk registers."""
        logging.info("Conducting stakeholder workshops for risk
prioritization...")
        # Simulate updating compliance documentation for NIST SP 800-53
        logging.info("Updated system inventory and risk register for
CMMC compliance.")

class Protect(NIST_CSF):
    """PROTECT: Zero-trust enforcement and automated patch
management."""

    def __init__(self, assets: List[str]):
        self.assets = assets

    def execute(self):
        self.automate()
```

```
        self.manage_change()

    def automate(self):
        """Deploys zero-trust policies and patches via automation."""
        logging.info("Enforcing zero-trust policies for remote work
access...")
        # Simulate automated patch deployment for legacy systems
        for asset in self.assets:
            if "Legacy" in asset:
                logging.info(f"Applying critical patch to {asset} via
automated system")

    def manage_change(self):
        """Trains workforce on zero-trust principles."""
        logging.info("Launching upskilling program for zero-trust
architecture...")
        # Simulate policy updates aligned with federal guidelines
        logging.info("Updated access control policies published to all
departments.")

class Detect(NIST_CSF):
    """DETECT: AI-driven monitoring for IoT/OT threats."""

    def __init__(self):
        self.threats = []

    def execute(self):
        self.automate()
        self.manage_change()

    def automate(self):
        """Uses AI/ML for real-time anomaly detection in OT systems."""
```

```
        logging.info("Deploying AI models to monitor energy grid IoT
devices...")
        # Simulate threat detection
        self.threats = ["Anomalous_Login_Attempt",
"OT_Protocol_Anomaly"]
        logging.warning(f"Detected threats: {self.threats}")

    def manage_change(self):
        """Collaborates with OT teams for threat validation."""
        logging.info("Conducting cross-functional red team/blue team
exercises...")
        # Simulate feedback loop integration
        logging.info("Updated detection rules based on OT team input.")

class Respond(NIST_CSF):
    """RESPOND: SOAR-driven incident containment."""

    def __init__(self, threats: List[str]):
        self.threats = threats

    def execute(self):
        self.automate()
        self.manage_change()

    def automate(self):
        """Triggers SOAR playbooks for automated containment."""
        logging.info("Executing SOAR playbook for ransomware
containment...")
        # Simulate automated isolation of compromised systems
        for threat in self.threats:
            logging.info(f"Isolating affected systems: {threat}")
```

```
    def manage_change(self):
        """Runs incident response simulations."""
        logging.info("Conducting tabletop exercises with executive
stakeholders...")
        # Simulate post-incident process improvements
        logging.info("Updated incident response plan published post-
exercise.")

class Recover(NIST_CSF):
    """RECOVER: Cloud-based disaster recovery and automated
rollback."""

    def __init__(self):
        self.recovery_status = {}

    def execute(self):
        self.automate()
        self.manage_change()

    def automate(self):
        """Orchestrates cloud recovery and system rollback."""
        logging.info("Initiating cloud-based recovery for healthcare
systems...")
        # Simulate automated rollback to pre-ransomware state
        self.recovery_status = {"EHR_System": "Recovered", "Downtime":
"45min"}
        logging.info(f"Recovery metrics: {self.recovery_status}")

    def manage_change(self):
        """Implements lessons learned from recovery efforts."""
        logging.info("Publishing post-incident review to improve
resilience...")
```

```
        # Simulate compliance updates
        logging.info("Updated backup policies to meet NIST SP 800-53
rev5.")

class CybersecurityStrategy:
    """Integrates all CSF functions with automation and change
management."""

    def __init__(self):
        self.identify = Identify()
        self.protect = Protect(self.identify.assets)
        self.detect = Detect()
        self.respond = Respond(self.detect.threats)
        self.recover = Recover()

    def execute_strategy(self):
        """Executes the full cybersecurity strategy lifecycle."""
        logging.info("**** Initiating NIST CSF Strategy Execution
****")
        self.identify.execute()
        self.protect.execute()
        self.detect.execute()
        self.respond.execute()
        self.recover.execute()
        logging.info("**** Strategy Execution Complete ****")

if __name__ == "__main__":
    # Simulate SolarWinds-style supply chain attack mitigation
    strategy = CybersecurityStrategy()
    strategy.execute_strategy()

    # Compliance validation check
```

```
    logging.info("Validating compliance with CMMC and NIST SP 800-
53...")
    # Simulate audit checks
    logging.info("Compliance validation completed successfully.")
```